

工业相机 SDK 二次开发示例程序说明（C#版）

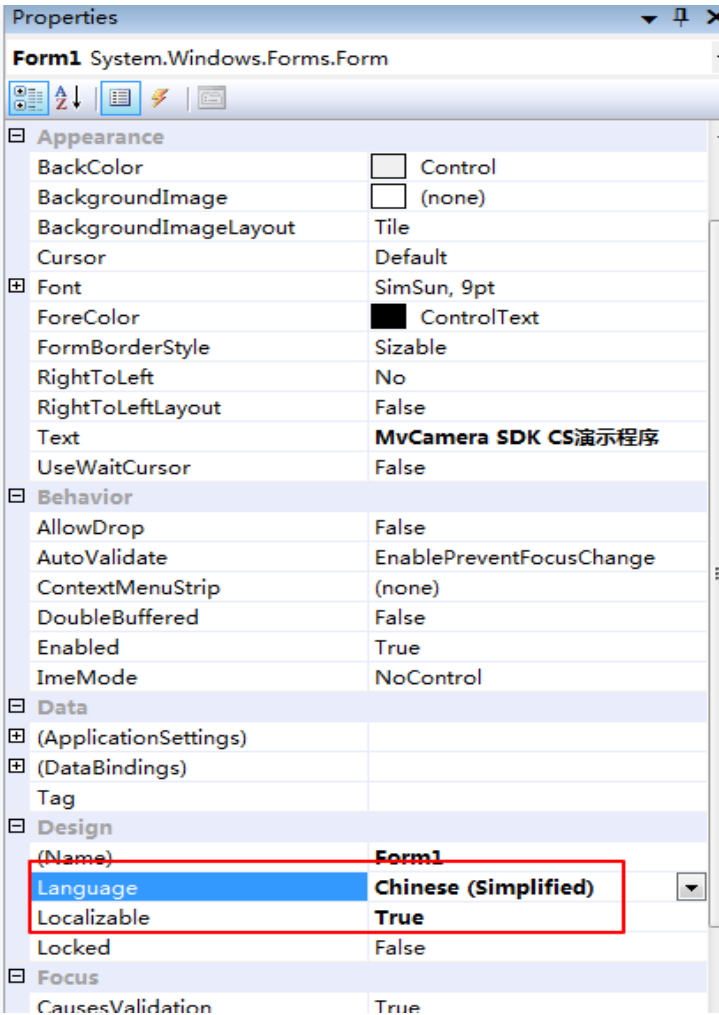
【摘要】

本文档主要介绍了使用工业相机 SDK(Software Development Kit)开发 C#程序方法及过程。在 SDK 开发包目录下,提供了 24 个 C#示例程序,其中 Form 程序 6 个,分别为 BasicDemo、ReconnectDemo、SetIODemo、ForceIpDemo、MultipleDemo、BasedOnGenTL; 控制台程序 18 个,分别为 CamLBasicDemo、ChunkData、ConnectSpecCamera、ConvertPixelFormat、Events、Grab_ActionCommand 、 Grab_Callback 、 GrabImage 、 GrabStrategies 、 MultiCast 、 ParametrizeCamera_FileAccess 、 ParametrizeCamera_LoadAndSave 、 Recording 、 SavePonitCloudData_3D 、 ImageEnhance 、 SpatialDenoise 、 LensShadingCorrection 和 ColorCorrect。这些示例程序分别从不同角度展示了利用 MvCameraControl.Net 进行开发的方法。

本文档就这六个 C# Form 示例程序的操作方法和开发流程展开讨论,介绍各个示例程序的使用步骤和开发流程,方便用户快速入门使用 C# SDK。

【注意】

C#版示例程序兼容中英文,对关键的程序会有中英文的注释,且界面控件也有中英文的区分,可通过切换属性的 **language** 实现。目前是默认打开以中文界面展现,exe 程序随系统语言而改变,用户可根据实际需求选择语言。如下所示,根据 form 界面的语言选择来进行转换。



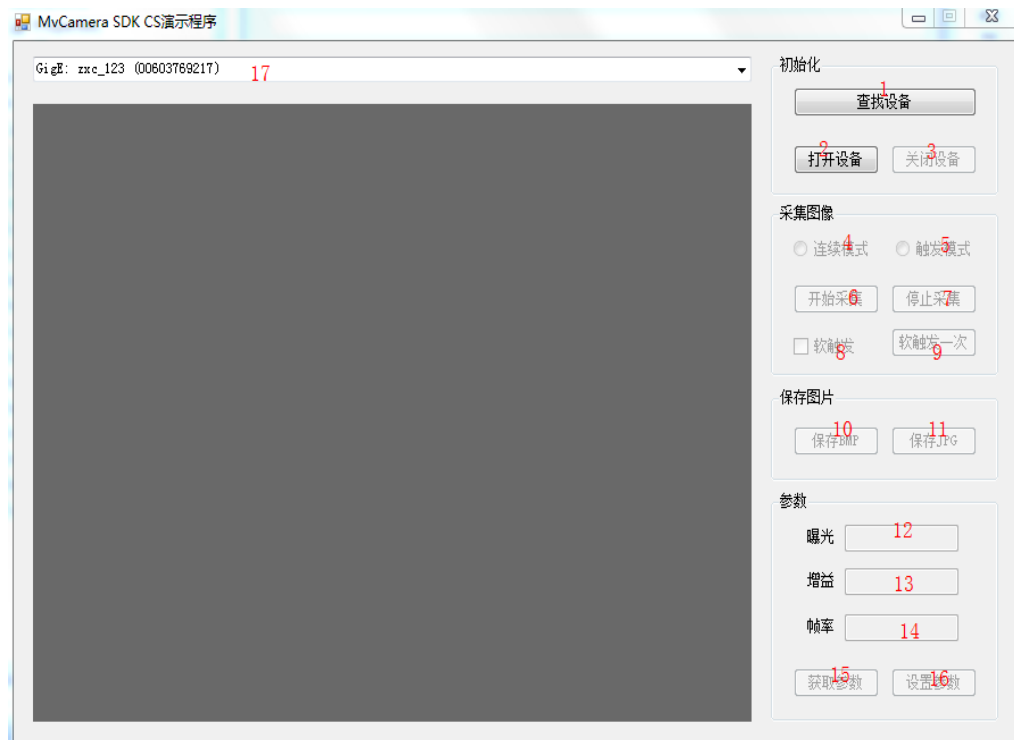
一. BasicDemo 使用步骤及开发流程

BasicDemo 是一个基本示例程序，包含了 SDK 使用过程中常用的一些接口调用，初次使用工业相机 SDK 进行二次开发的用户推荐首先参考 BasicDemo，其涵盖了大多数用户对 SDK 的使用方法示例需求。

1.1 Demo 软件使用步骤

1.1.1 界面总体

软件界面总览，一共包括四个控制模块(初始化，图像采集，图片保存，参数控制)、一个下拉设备列表和一个图像显示区域



1.1.2 使用过程

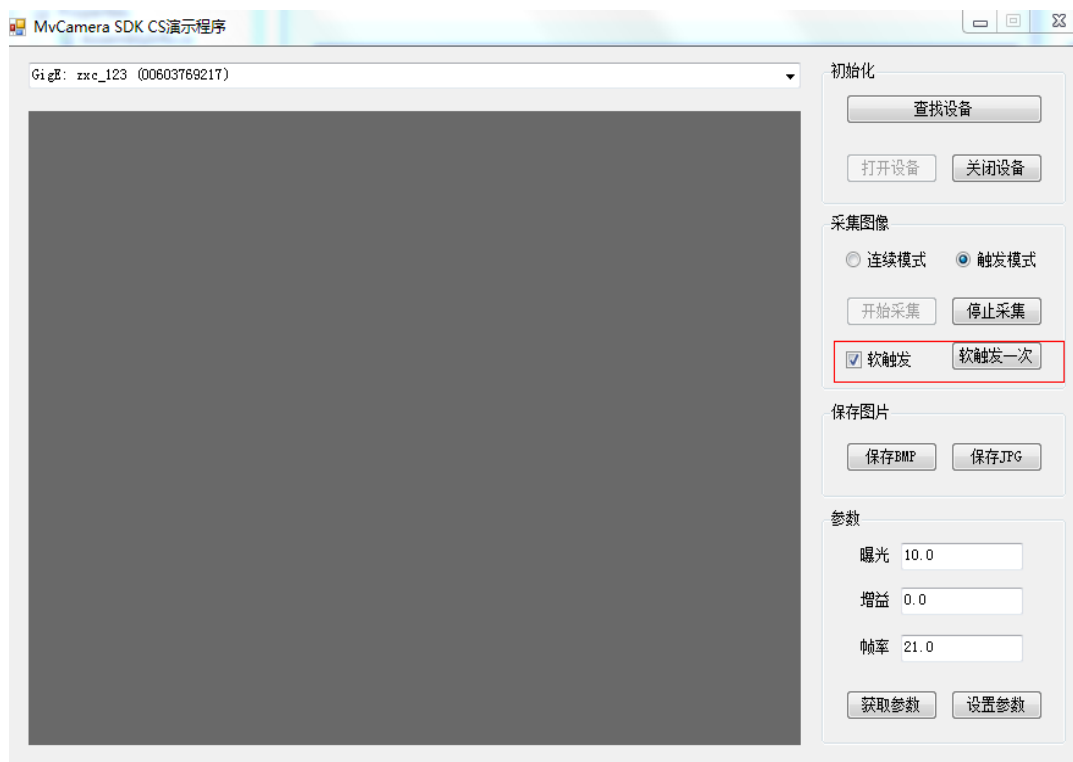
点击【查找设备】进行查找设备，这时（17）会出现当前在线的设备列表，命名方式为 用户 ID 不为空时显示设备类型+设备名称+序列号，ID 为空时显示设备类型+设备型号 + 序列号，选择其中一个设备



点击【打开设备】打开当前选中的设备，默认以连续方式打开设备。选择触发模式可以选中触发模式单选框。



在触发模式下，可以设置为软触发，当点击【开始采集】后，同时【软触发一次】也是可以点击从而完成触发一次功能



采用连续模式下，点击【开始采集】进行图像采集，左边的显示区域将会出现实时图像

此时，若点击【保存 BMP】或者【保存 JPG】，将会在当前 exe 目录下出现一个名称为 Image.bmp 或者 Image.jpg 的图片，即为保存的当前图像

点击【获取参数】将会刷新当前的曝光时间、增益和帧率的数值，而更改【曝光】、【增益】、【帧率】的数值之后点击【设置参数】将会重新设置新的曝光时间、增益和帧率的数值



在使用过程中有任何异常或错误，都会以弹窗的形式出现提示，若没有任何提示，则认为一切正常地运行

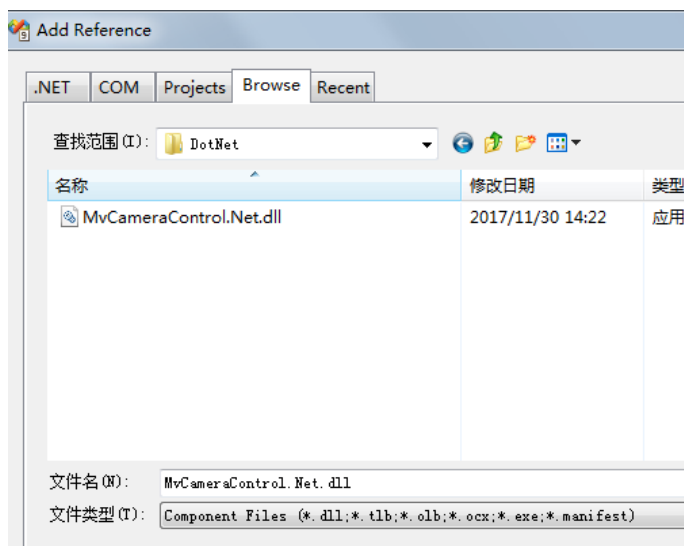
1.2 Demo 软件开发步骤

1.2.1 DLL 加载

安装好 MVS 的同时会把相应 32 和 64 的 dll 打到环境变量。

1.2.2 工程配置

创建 CS 工程并添加引用，加入 MvCameraControl.Net.dll 到工程中。



1.2.3 引用命名空间

添加引用后再工程中引用命名空间 `using MvCamCtrl.NET`, 就可以调 `MyCamera` 类中相机操作的函数。

```

1 .....public MyCamera();
2
3
4 .....public static object ByteToStruct(byte[] bytes, Type type);
5 .....public IntPtr GetCameraHandle();
6 .....public int MV_CC_CloseDevice_NET();
7 .....public int MV_CC_ConvertPixelType_NET(ref MyCamera.MV_PIXEL_CONVERT_PARAM pstCvtPa
8 .....public int MV_CC_CreateDevice_NET(ref MyCamera.MV_CC_DEVICE_INFO stDevInfo);
9 .....public int MV_CC_CreateDeviceWithoutLog_NET(ref MyCamera.MV_CC_DEVICE_INFO stDevIn
0 .....public int MV_CC_DestroyDevice_NET();
1 .....public int MV_CC_Display_NET(IntPtr hWnd);
2 .....public static int MV_CC_EnumDevices_NET(uint nTLayerType, ref MyCamera.MV_CC_DEVIC
3 .....public static int MV_CC_EnumerateTls_NET();
4 .....public int MV_CC_GetAcquisitionLineRate_NET(ref MyCamera.MVCC_INTVALUE pstValue);
5 .....public int MV_CC_GetAcquisitionMode_NET(ref MyCamera.MVCC_ENUMVALUE pstValue);
6 .....public int MV_CC_GetAllMatchInfo_NET(ref MyCamera.MV_ALL_MATCH_INFO pstInfo);
7 .....public int MV_CC_GetAOIoffsetX_NET(ref MyCamera.MVCC_INTVALUE pstValue);
8 .....public int MV_CC_GetAOIoffsetY_NET(ref MyCamera.MVCC_INTVALUE pstValue);
9 .....public int MV_CC_GetAutoExposureTimeLower_NET(ref MyCamera.MVCC_INTVALUE pstValue)
0 .....public int MV_CC_GetAutoExposureTimeUpper_NET(ref MyCamera.MVCC_INTVALUE pstValue)
1 .....public int MV_CC_GetBalanceRatioBlue_NET(ref MyCamera.MVCC_INTVALUE pstValue);
2 .....public int MV_CC_GetBalanceRatioGreen_NET(ref MyCamera.MVCC_INTVALUE pstValue);
3 .....public int MV_CC_GetBalanceRatioRed_NET(ref MyCamera.MVCC_INTVALUE pstValue);
4 .....public int MV_CC_GetBalanceWhiteAuto_NET(ref MyCamera.MVCC_ENUMVALUE pstValue);
5 .....public int MV_CC_GetBoolValue_NET(string strKey, ref bool pbValue);
6 .....public int MV_CC_GetBrightness_NET(ref MyCamera.MVCC_INTVALUE pstValue);
7 .....public int MV_CC_GetBurstFrameCount_NET(ref MyCamera.MVCC_INTVALUE pstValue);
8 .....public int MV_CC_GetDeviceInfo_NET(ref MyCamera.MV_CC_DEVICE_INFO pstDevInfo);
9 .....public int MV_CC_GetDeviceUserID_NET(ref MyCamera.MVCC_STRINGVALUE pstValue);
0 .....public int MV_CC_GetEnumValue_NET(string strKey, ref MyCamera.MVCC_ENUMVALUE pstVa
1 .....public int MV_CC_GetExposureAutoMode_NET(ref MyCamera.MVCC_ENUMVALUE pstValue);
2 .....public int MV_CC_GetExposureTime_NET(ref MyCamera.MVCC_FLOATVALUE pstValue);
3 .....public int MV_CC_GetFloatValue_NET(string strKey, ref MyCamera.MVCC_FLOATVALUE pst
4 .....public int MV_CC_GetFrameRate_NET(ref MyCamera.MVCC_FLOATVALUE pstValue);
5 .....public int MV_CC_GetGain_NET(ref MyCamera.MVCC_FLOATVALUE pstValue);
6 .....public int MV_CC_GetGainMode_NET(ref MyCamera.MVCC_ENUMVALUE pstValue);
7 .....public int MV_CC_GetGamma_NET(ref MyCamera.MVCC_FLOATVALUE pstValue);

```

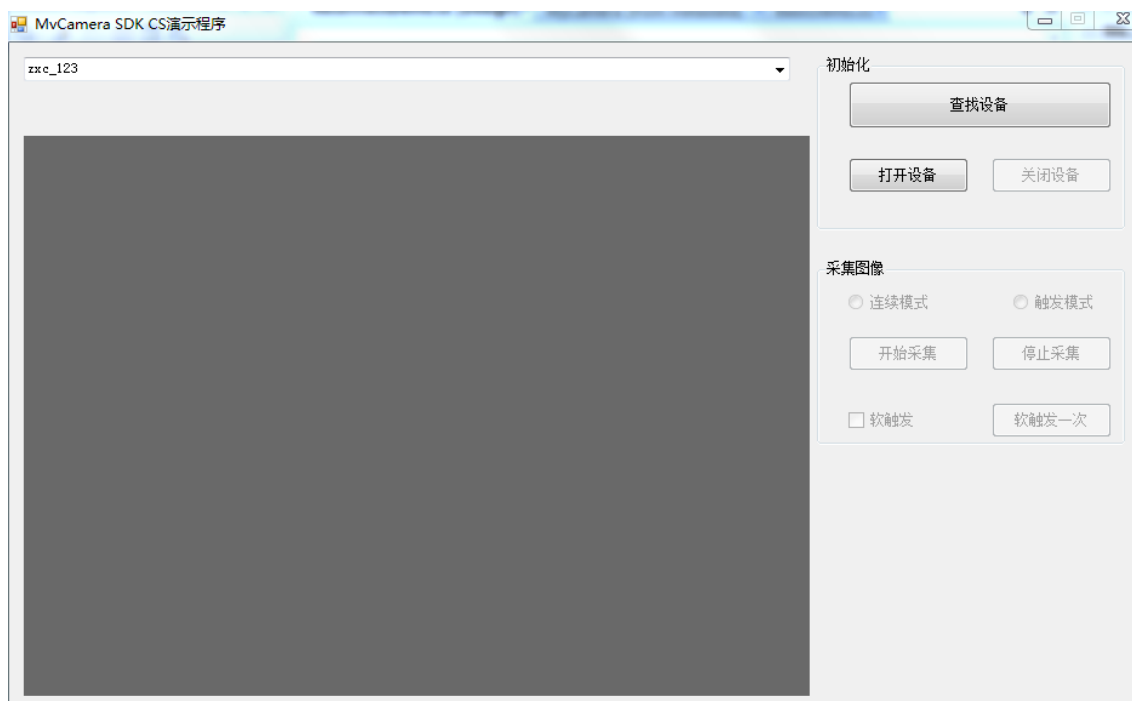
二. ReconnectDemo 使用步骤及开发流程

ReconnectDemo 重点展示了 SDK 中相机断线重连的操作步骤。告知用户如何使用断线回调以及如何重新连接相机。

2.1 Demo 软件使用步骤

2.1.1 界面总体

总体界面如下图。界面类似 **BasicDemo**，具有查找设备、打开设备、关闭设备、开始采集、停止采集、设置触发等功能。



2.1.2 使用过程

ReconnectDemo 中，当相机断线时，程序会进入异常回调，异常回调中，会根据当前选中的相机信息进行不断的尝试连接，当相机在线时则会被连接上。

2.2 Demo 软件开发步骤

关于相机操作的开发流程与 **BasicDemo** 相似。本节重点介绍回调函数的使用方法。

在 C# 中，用 `delegate`（代理）的方式代替 C 语言中函数指针。在工业相机 C# SDK 中，异常断线的回调代理为 `MyCamera.cbExceptiondelegate`。

首先在 `Form1` 类中申明一个回调代理成员变量，如下：

```
MyCamera.cbExceptiondelegate pCallBackFunc;
```

然后为 `pCallbackFunc` 创建一个实例：

```
pCallbackFunc= new MyCamera. cbExceptiondelegate (cbExceptiondelegate);
```

其中，`cbExceptiondelegate` 表示回调处理函数。

其次，在打开相机操作之后，利用 SDK 中注册回调函数接口，注册回调函数。当相机异常断线时，程序会进入异常回调。用户可在异常回调中进行重新连接相机的操作。注册过程如下：`m_pMyCamera.MV_CC_RegisterExceptionCallBack_NET(pCallbackFunc, IntPtr.Zero);`

在本示例程序中，`cbExceptiondelegate` 函数先会对进行 `CloseDevice` 和 `DestroyHandle` 操作，之后则会不断的尝试连接相机。

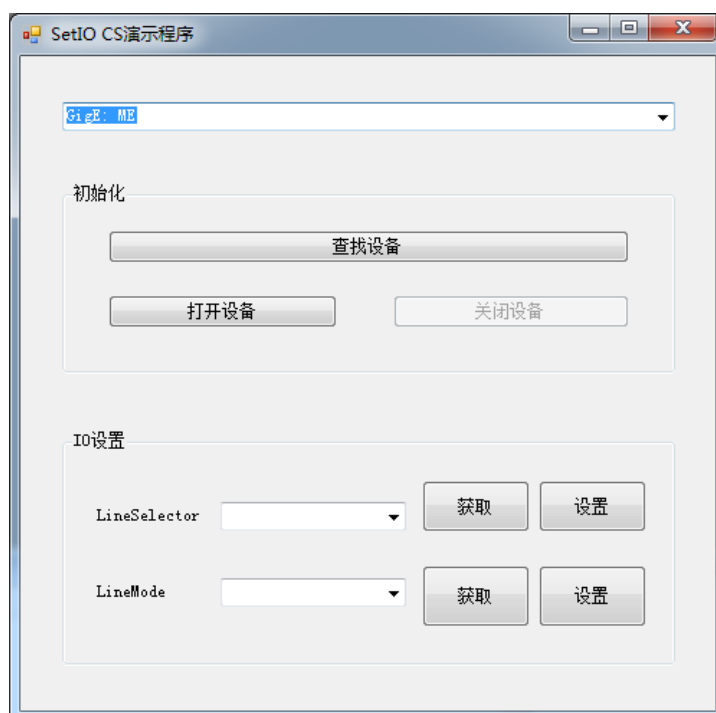
三. SetIODemo 使用步骤及开发流程

本节介绍的 Demo 主要实现对相机 IO 输入输出的控制。使用用户群体为需要对相机 IO 进行控制的用户。

当用户需要使用功能相机 IO 属性时，首先需要将相机设置成触发模式，并且选择相应的触发源 `TriggerSource`，例如选择 `Line0` 进行输入设置，可以选择高电平、低电平触发等；然后在 `Digital IO Control` 中对触发源进一步设置，比如滤波，延时等，也可以对 `Line1` 进行输出设置，对 `Line2` 进行输入或者输出的设置；其中输入和输出对应不同颜色的信号线，不同系列的相机，IO 定义和接线也可能不同，所以设置好这些 IO 属性后，要通过相机 IO 特定的接线图，连接对应的信号线，来实现该 IO 的功能。本节介绍的 demo 是关于相机 IO 属性的简单设置，相机的 `Digital IO Control` 详细设置可以参考 MVS。

3.1 Demo 软件使用步骤

3.1.1 界面总体



3.1.2 使用过程

相机基本操作与 BasicDemo 相似。打开一个设备后可以对相机的 IO 属性进行获取和设置。IO 属性主要有 LineSelector 和 LineMode 两个。分别点击获取和设置可以对相应的属性进行读取和写入。

3.2 Demo 软件开发步骤

3.2.1 IO 属性

有关相机 IO 属性主要有两个:LineSelector 和 LineMode。LineSelector 指输出端口选择，目前相机主要有三个 IO 端口：Line0，Line1，Line2。其中，Line0 只可配置为输入，Line1 只可配置为输出，Line2 可配置为输入或者输出。LineMode 表示输入或者输出模式。

3.2.2 获取和设置接口

在示例程序中，获取和设置 IO 用到的接口分别为：
`m_pMyCamera.MV_CC_GetEnumValue_NET(string strKey, ref CSI.MVCC_ENUMVALUE pstValue)`，以及 `m_pMyCamera.MV_CC_SetEnumValue_NET(string strKey, UInt32 nValue)`

在 SDK 中，类似此类 Set 或 Get + 数据类型 + Value 的接口函数成为万能接口函数，其作用为获取或设置相机任意属性值。万能接口的第一个参数为属性名称，为一个 string 型字符串，相机属性名称可以通过 MVS 客户端的属性树查询。第二个参数为获取到的或者设置的属性值。

3.2.3 IO 操作

在本节示例程序中，主要用到的属性节点为”LineSelector”以及”LineMode”，其属性类型均为 Enumeration 类型。调用万能接口即可实现对其属性的操作。

获取操作如下：

```
MyCamera.MVCC_ENUMVALUE stSelValue = new
    MyCamera.MVCC_ENUMVALUE();
nRet = m_pMyCamera.GetEnumValue("LineSelector", ref stSelValue);
MyCamera.MVCC_ENUMVALUE stModeValue = new
    MyCamera.MVCC_ENUMVALUE();
nRet = m_pMyCamera.GetEnumValue("LineMode", ref stModeValue);
```

设置操作如下：

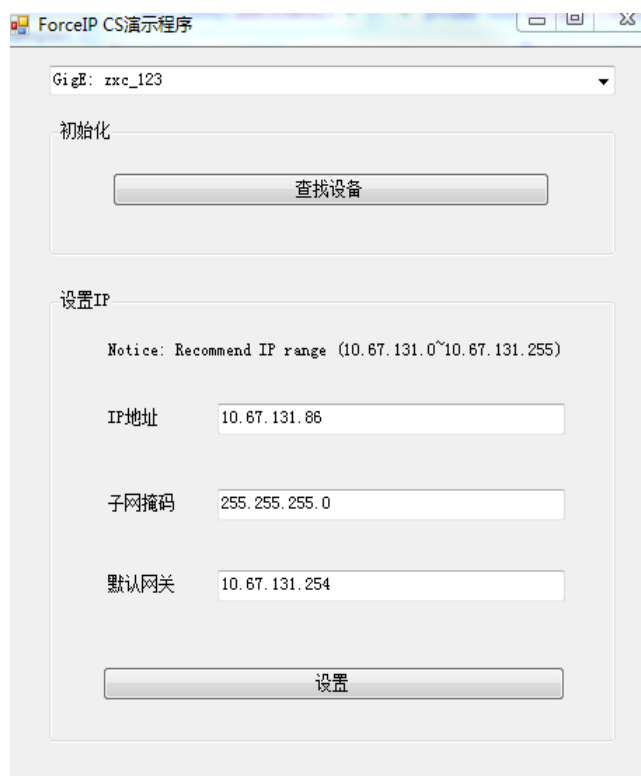
```
nRet = m_pMyCamera.SetEnumValue("LineSelector", nValue);
nRet = m_pMyCamera.SetEnumValue("LineMode", nValue);
```

四. ForceIpDemo 使用步骤和开发流程

4.1 Demo 软件使用步骤

4.1.1 界面总体

软件界面如下图所示。



界面主要分为两个模块：初始化模块和设置 IP 模块。

4.1.2 使用过程

首先，点击查找设备对网段内的设备进行枚举，软件自动选择列表中第一项。

然后，选择需要配置 IP 的设备。

在设置 IP 模块的提示信息中会提示本机网卡所在的网段并显示建议设置的 IP 范围。在输入框中输入想要设置的 IP，点击设置。

4.2 Demo 软件开发步骤

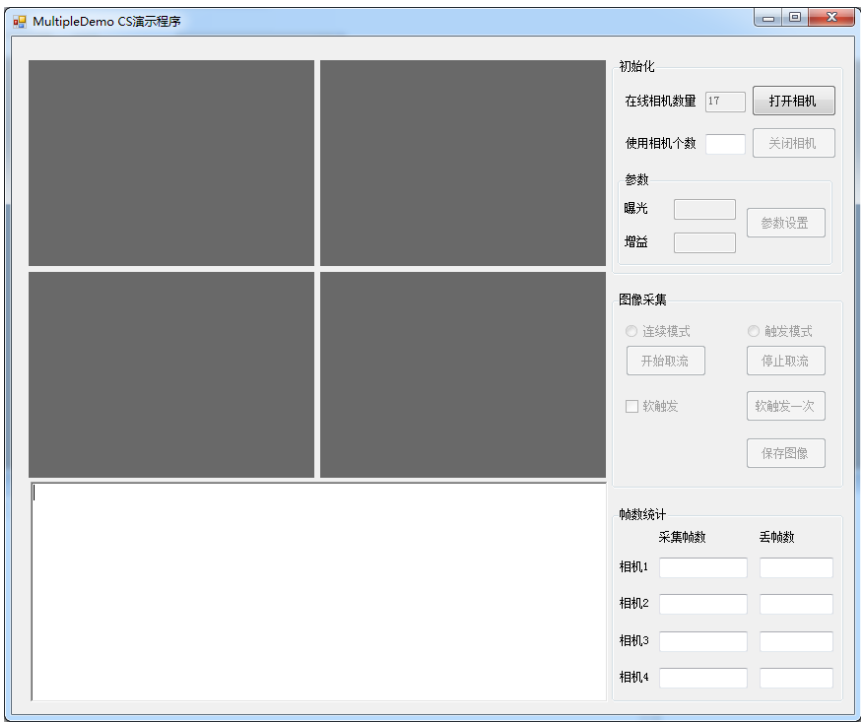
设置 IP 调用 SDK 中 [MyCamera.MV_GIGE_ForceIp_NET\(UInt32 nIp\)](#)接口。

五. MultipleDemo 使用步骤及开发流程

5.1 Demo 软件使用步骤

5.1.1 界面总体

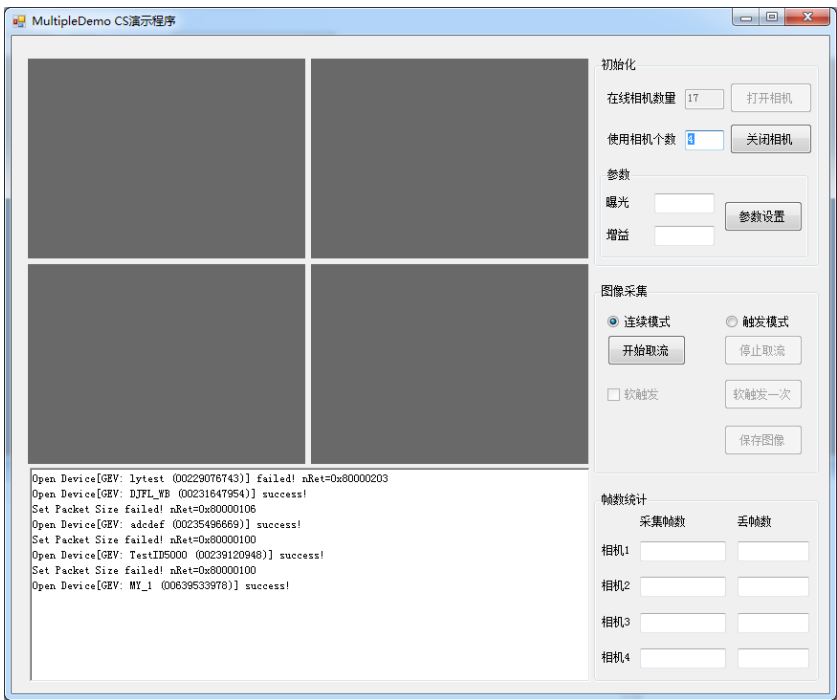
总览界面,软件界面主要包括三个控制模块(初始化、参数设置、采集图像)，四块图像显示区域以及帧数信息显示区域。



5.1.2 使用过程

打开软件，“在线设备数量”会自行枚举在线相机个数，在“使用设备个数”文本框内填写需要打开的相机个数 n，单击“打开相机”，默认以连续方式打开 n 台设备。

在“曝光”和“增益”中填写修改的参数，单击“设置参数”，即可依次修改 n 台设备参数。同时可选择连续或者触发模式。



点击“开始采集”，左侧会显示预览图像。同时采集帧数和丢帧数会即时更新数据（1秒更新一次）。此时若点击“保存图片”，会在当前 exe 目录下出现一个名称为 image1-image4 的 bmp 文件，分别对应 1-4 台设备保存的图片。若希望结束，则点击“停止采集”，“关闭设备”即可。

当出现异常和错误时，会在输出框内进行显示。

5.2 Demo 软件开发步骤

5.2.1 多相机的实现

MultipleDemo 在 BasicDemo 基础上，在类中添加 `m_bEnabled` 数组的成员变量，表示四台相机的使能，初始化时由“使用数量”和是否成功打开决定 `m_bEnabled` 为 True 或者 False。后面的基本操作均由 `m_bEnabled` 判断是否需要对应的相机进行操作。

5.2.2 总帧数、丢帧数、保存图片

总帧数在回调函数中计数（成员变量）。回调函数中同时完成保存图片的功能，判断是否点击保存图片的按钮确定是否保存当前帧为图片，保存完成后，修改相应标志位以免下次取流重复保存图片。丢帧数由调用 `CSI.MV_CC_GetAllMatchInfo_CSI(ref pstInfo)` 接口获取。总帧数和丢帧数的更新周期为 1 秒，设置定时器，1 秒获取一次丢帧数，然后再更新总帧数和丢帧数。

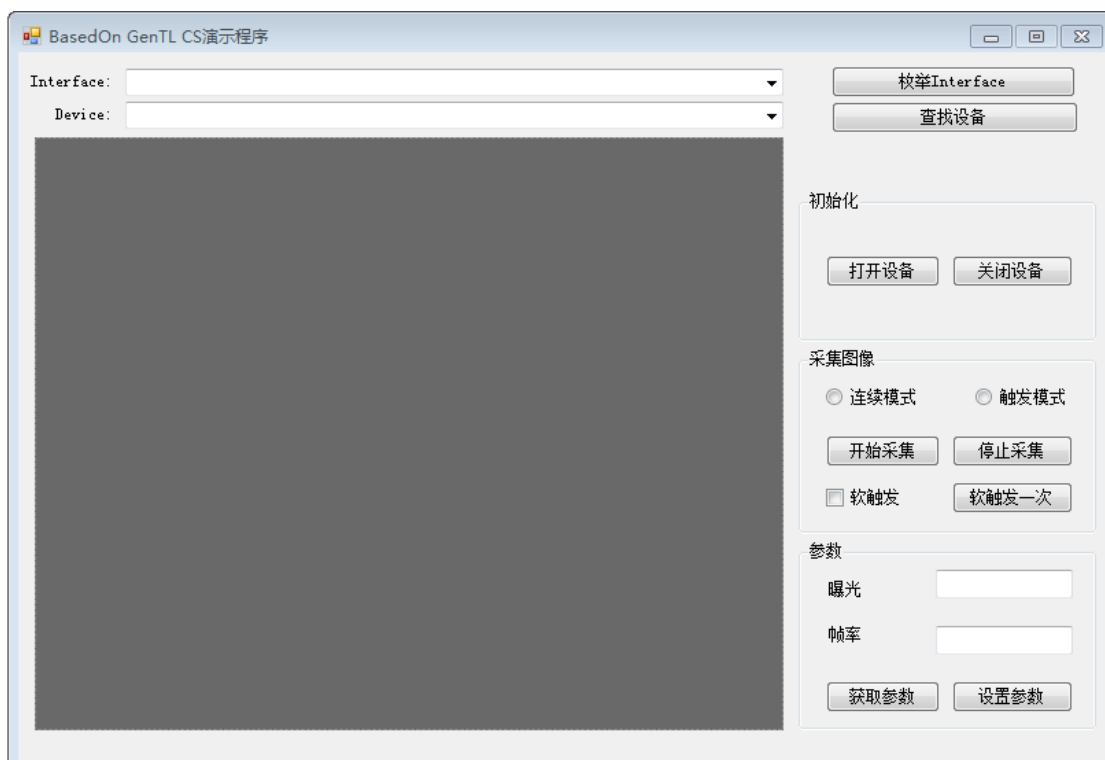
六. BasedOnGenTL 使用步骤及开发流程

BasedOnGenTL 是一个使用 GenTL 的示例程序，可以加载不同的 CTI 文件，枚举到对应的设备，其他操作类似于 BasicDemo。

6.1 Demo 软件使用步骤

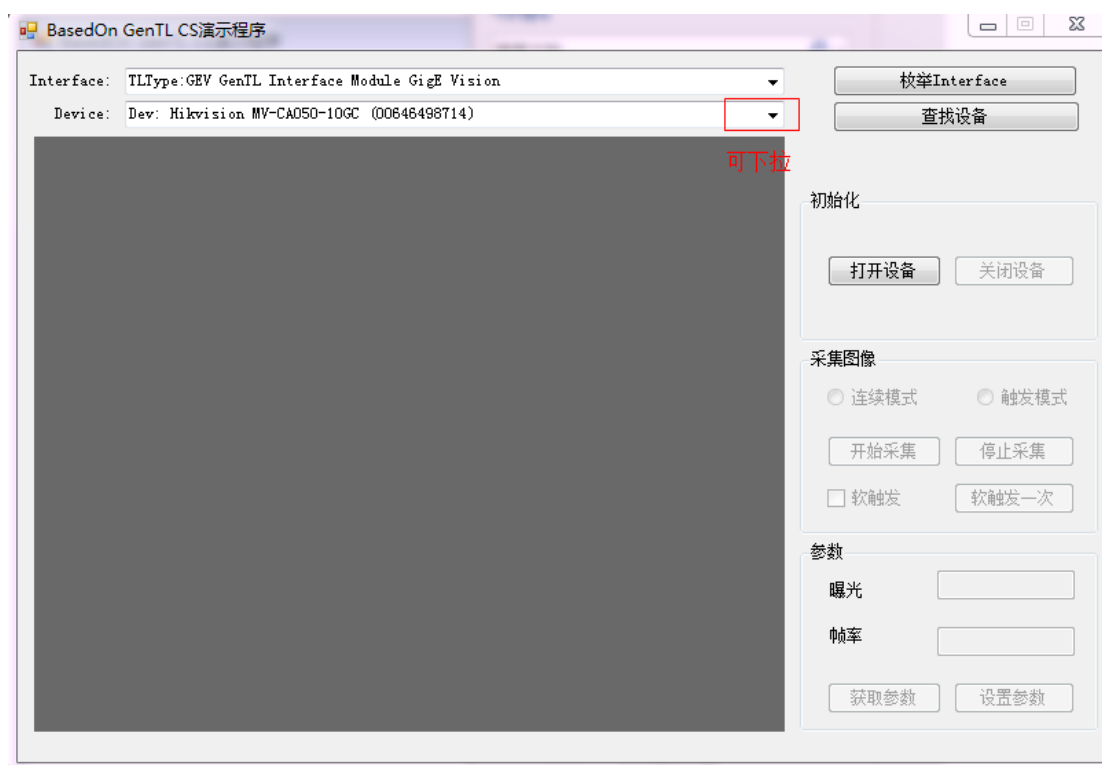
6.1.1 界面总体

软件界面总览，一共包括五个控制模块(枚举 `Interface`、查找设备、初始化，图像采集，参数控制)、一个下拉设备列表和一个图像显示区域；

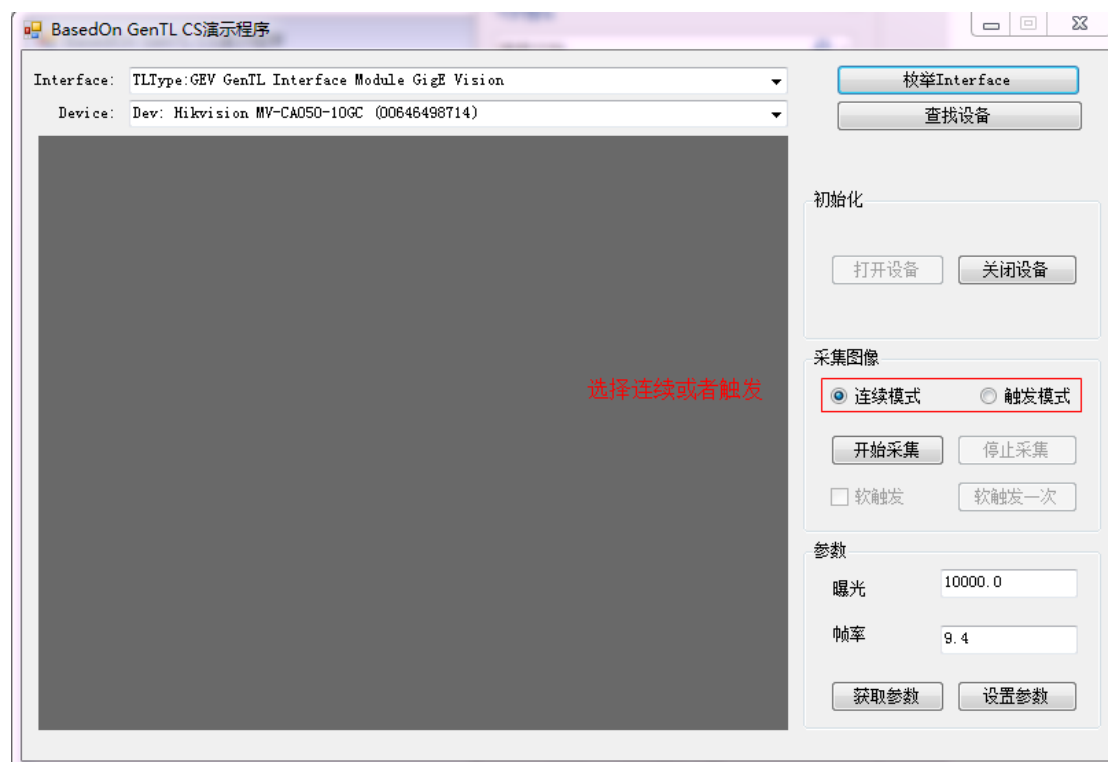


6.1.2 使用过程

首先点击【枚举 Interface】，选择需要的 CTI 文件，例如选择 MVS 安装包路径 C:\Program Files (x86)\Common Files\MVS\Runtime\Win32_i86 下的 MvProducerGEV.cti 文件，然后点击【查找设备】进行查找设备，这时下拉列表会出现当前在线 GigeVision 设备列表，命名方式为用户 ID 不为空时显示设备类型+设备名称+IP 地址，设备为空时显示设备类型+设备型号+IP 地址。选择不同的 CTI 文件，会枚举出不同类型的设备。选择其中一个设备；

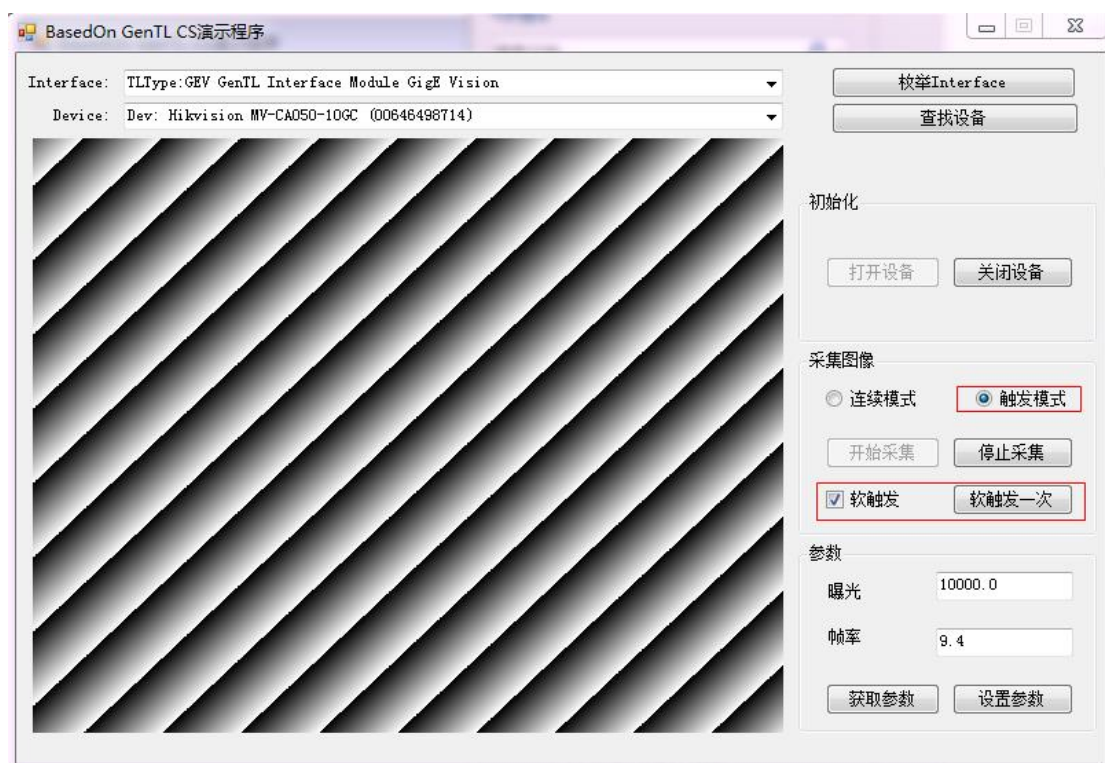


点击【打开设备】打开当前选中的设备，默认以连续方式打开设备。选择触发模式可以选中触发模式单选框。

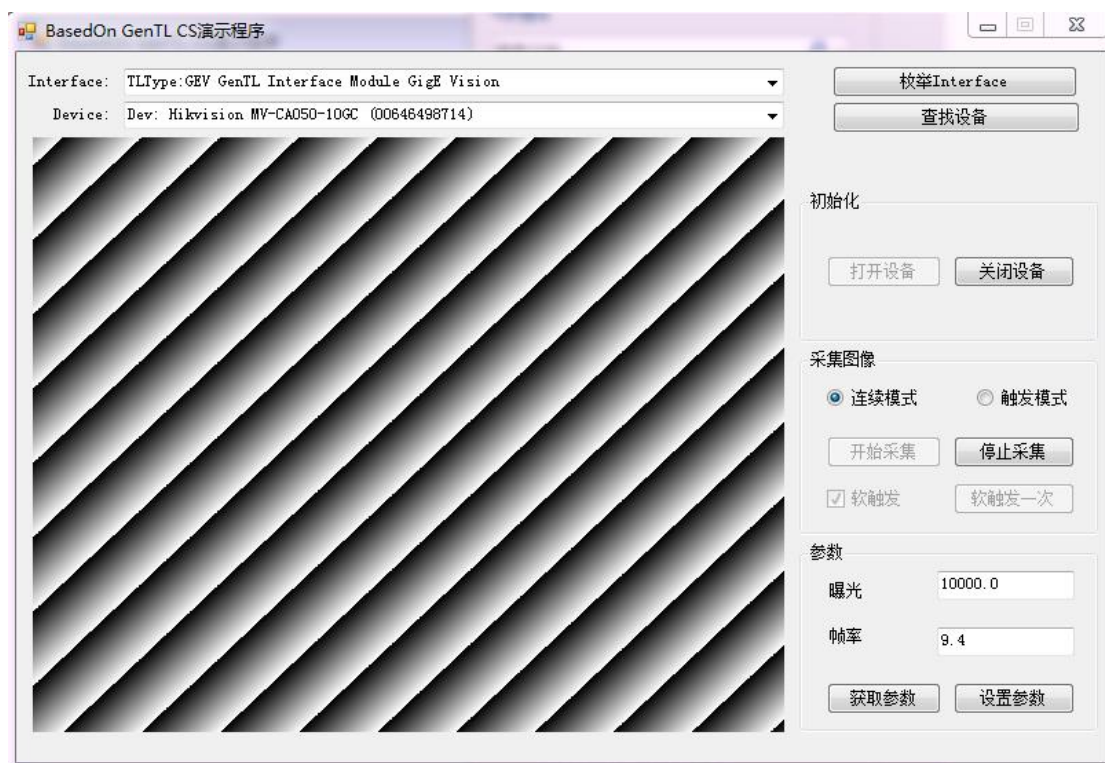


在触发模式下，可以设置为软触发，当点击【开始采集】后，同时【软触发一次】也是

可以点击从而完成触发一次功能



采用连续模式下，点击【开始采集】进行图像采集，左边的显示区域将会出现实时图像
点击【获取参数】将会刷新当前的曝光时间、帧率的数值，而更改【曝光】、【帧率】的数值之后点击【设置参数】将会重新设置新的曝光时间、帧率的数值



在使用过程中有任何异常或错误，都会以弹窗的形式出现提示，若没有任何提示，则认

为一切正常地运行

6.2 Demo 软件开发步骤

同 1.2 节 Demo 开发步骤。